

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-265747

(43)Date of publication of application : 28.09.2001

(51)Int.Cl.

G06F 15/16
G06F 9/44
G06F 13/00
G06F 15/177

(21)Application number : 2000-074273

(71)Applicant : HITACHI LTD

(22)Date of filing : 16.03.2000

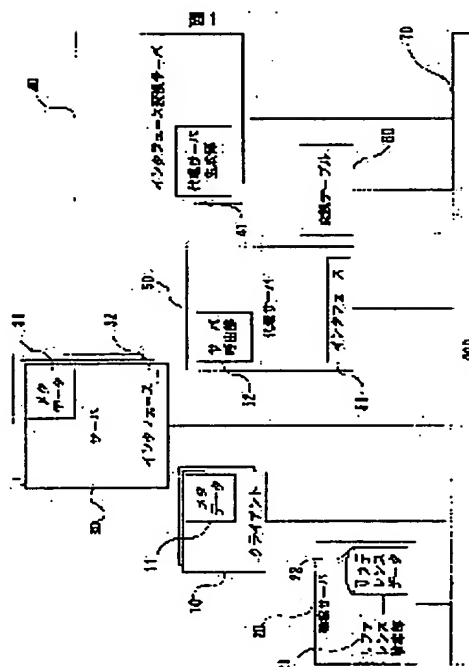
(72)Inventor : SHIGEOKA TOMOHIKO

(54) DISTRIBUTED OBJECT LINKING DEVICE

(57)Abstract:

PROBLEM TO BE SOLVED: To enable the change of an interface to be utilized by a client by providing a proxy server for connecting the client and a server.

SOLUTION: Concerning a distributed object linking device, with which an interface 32 to be utilized by a server is not matched with an interface to be utilized by a client, composed of plural servers 31 for providing services and a client 10 to utilize the server, an interface converting server 40 generates an interface converting table 60 for matching the interface to be utilized by the client and the interface to be utilized by the server and a proxy server 50 for connecting the client and the server while referring to the interface converting table and provides services through the generated proxy server.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

BEST AVAILABLE COPY

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁(JP)

(12)公開特許公報 (A)

(11)特許出願公開番号

特開 2001-265747

(P2001-265747A)

(43)公開日 平成13年9月28日(2001.9.28)

(51)Int. Cl. ⁷	識別記号	F I	テ-マ-ト*(参考)
G 0 6 F	15/16	6 2 0	T 5B045
	9/44	5 3 0	M 5B089
	13/00	3 5 7	Z
	15/177	6 7 6	H

審査請求 未請求 請求項の数 5 O L

(全 8 頁)

(21)出願番号 特願2000-74273(P2000-74273)

(22)出願日 平成12年3月16日(2000.3.16)

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目四番地

(72)発明者 茂岡 知彦

神奈川県横浜市戸塚区戸塚町5030番地 株

式会社日立製作所ソフトウェア事業部内

(74)代理人 100078134

弁理士 武 顕次郎

Fターム(参考) 5B045 AA00 BB11 BB28 BB47 BB58

GG09

5B089 GA11 GA21 GB08 GB10 JA11

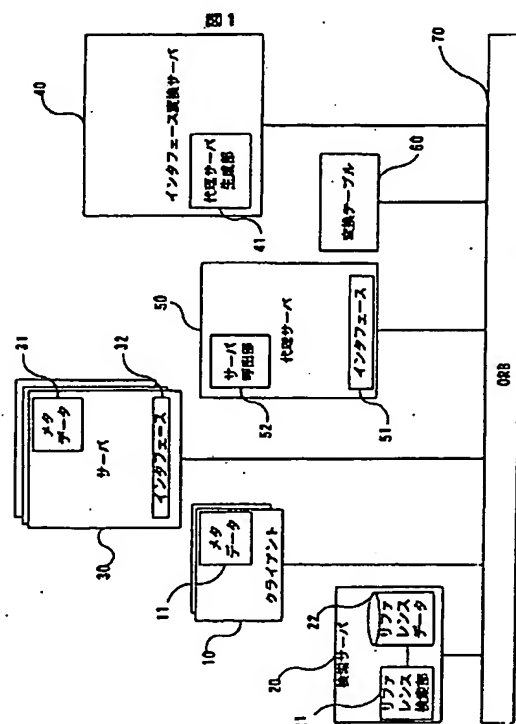
JB22 KC15 KH05 KH30

(54)【発明の名称】分散オブジェクト連携装置

(57)【要約】

【課題】クライアントおよびサーバ間を接続する代理サーバを備えることにより、クライアントが利用するインタフェースを変更可能にする。

【解決手段】サービスを提供する複数のサーバ31とサーバを利用するクライアント10からなり、前記サーバが利用するインタフェース32とクライアントが利用するインタフェースが整合しない分散オブジェクトの連携装置において、インタフェース変換サーバ40は、クライアントの利用するインタフェースとサーバの利用するインタフェースとを整合するインタフェース変換テーブル60および前記インタフェース変換テーブルを参照して前記クライアントおよびサーバ間を接続する代理サーバ50を生成し、生成した代理サーバを介してサービスを提供する。



【特許請求の範囲】

【請求項1】 サービスを提供する複数のサーバとサーバを利用するクライアントからなり、前記サーバが利用するインターフェースとクライアントが利用するインターフェースが整合しない分散オブジェクトの連携装置において、

クライアントの利用するインターフェースとサーバの利用するインターフェースとを整合するインターフェース変換テーブルおよび前記インターフェース変換テーブルを参照して前記クライアントおよびサーバ間を接続する代理サーバを備え、該代理サーバを介してサービスを提供することを特徴とする分散オブジェクトの連携装置。

【請求項2】 請求項1の記載において、前記サーバが利用するインターフェースとクライアントが利用するインターフェースをもとに前記インターフェース変換テーブルを生成するインターフェース変換サーバを備えたことを特徴とする分散オブジェクトの連携装置。

【請求項3】 請求項1ないし請求項2の何れか1の記載において、前記代理サーバを生成するインターフェース変換サーバを備えたことを特徴とする分散オブジェクトの連携装置。

【請求項4】 請求項2ないし請求項3の何れか1の記載において、前記インターフェース変換サーバはクライアントが利用するインターフェース情報をクライアントが利用するサーバを介して取得することを特徴とする分散オブジェクトの連携装置。

【請求項5】 請求項2ないし請求項4の何れか1の記載において、クライアントが利用するインターフェース情報は、前記クライアントが利用するサーバが解読不能に暗号化したことを特徴とする分散オブジェクトの連携装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は分散オブジェクトの連携装置にかかり、特にネット上に分散配置したサーバを有効に利用することのできる分散オブジェクトの連携装置に関する。

【0002】

【従来の技術】特開平10-254701号公報には、分散オブジェクトシステムにおいて、利用者とサービス提供者間のインタフェースの不整合を解消する方法が示されている。この方法においては、利用者は予め定義された抽象インターフェースを持つサービスにアクセスすることにより、前記抽象インターフェースと、実サービスのインタフェースとの間の相違をインタフェース整合機構により補正している。

【0003】また、分散オブジェクトシステムにおいて、インタフェース仕様を管理する仕組みとしては、オブジェクト・マネジメント・グループ(Object Management Group: OMG)の1999年6月第2.3版 (Revisi

on 2.3, June 1999) のCORBA仕様記載のインタフェース・レポジトリ(Interface Repository: IR)がある。このIRにはインタフェースに関する情報を提供するメソッドを備えたオブジェクトの階層が入っていて、インタフェース仕様の参照や更新を行うことができる。

【0004】また、オブジェクトに動的な要求の構成を行う仕組みとしてのCORBA仕様には、動的起動インタフェース(Dynamic Invocation Interface: DII)が記載されている。このDIIを使用することで、クライアントをコンパイルした時点ではインタフェースがないオブジェクトも含めて、任意のオブジェクトを起動できるクライアントを作成できる。

【0005】また、オブジェクトが動的に要求を受け付けるための仕組みとしてのCORBA仕様には動的スケルトンインタフェース(Dynamic Skelton Interface: DSI)が記載されている。このDSIを使用すると、サーバオブジェクトはコンパイル時に定義されていないオブジェクトへのクライアントオペレーションリクエストをディスパッチ可能である。

【0006】

【発明が解決しようとする課題】前記特開平10-254701号公報に示される技術においては、クライアントが呼び出せるのは定義されている抽象インタフェースのみであり、また、クライアントはそのインタフェースを呼び出すように作られていなければならない。すなわち、クライアントの製作時点においてクライアントが利用する抽象インタフェースの定義が参照可能でなければならない。また、利用する実サービス提供オブジェクトが変更される毎に、インタフェースを変換する変換モジュールを再生成しなければならない。

【0007】さらに、前記CORBA仕様記載のIRはインタフェースの構文的仕様のみを扱うものであり、インタフェースの意味的仕様を扱うものではない。また、CORBA仕様記載のDIIおよびDSIは通常のインタフェースより利用方法が複雑であり、分散オブジェクトの呼び出しに通常の関数呼び出しと異なる記述を必要とする特殊な仕組みである。

【0008】本発明は前記問題点を鑑みてなされたもので、クライアントが利用するインタフェースを定義済みのインタフェース以外に変更することができ、また、実行時の具体的なインタフェースを利用者およびサービス提供側が互いに未知の状態では通信可能な分散オブジェクトの連携装置を提供する。

【0009】

【課題を解決するための手段】本発明は、上記の課題を解決するために次のような手段を採用した。

【0010】サービスを提供する複数のサーバとサーバを利用するクライアントからなり、前記サーバが利用するインターフェースとクライアントが利用するインターフェースが整合しない分散オブジェクトの連携装置にお

いて、クライアントの利用するインターフェースとサーバの利用するインターフェースとを整合するインターフェース変換テーブル60および前記インターフェース変換テーブルを参照して前記クライアントおよびサーバ間を接続する代理サーバ50を備え、サーバは該代理サーバを介してクライアントにサービスを提供する。また、インタフェース変換サーバはクライアントの利用するインターフェースとサーバの利用するインターフェースをもとに前記変換テーブルおよび代理サーバを生成する。

【0011】さらに、クライアントはクライアントが利用するインターフェース情報を前記クライアントが利用するサーバが解読不能に暗号化して送信することができる。

【0012】

【発明の実施の形態】以下に本発明の第1の実施形態を図1ないし図9を用いて説明する。図1は本発明の第1の実施形態にかかる分散オブジェクトの連携装置を示す図である。図において、10はクライアント、11はクライアントのメタデータであり、クライアント自身が利用するインタフェースの意味的仕様および構文的仕様を表す。20は検索サーバであり、クライアントが利用するサーバをサーバ名称あるいは提供サービス名称から検索する。21はリファレンス検索部、22はサーバのリファレンス、サーバの名称および提供サービスの名称を格納したリファレンスデータベースである。30はサーバ、31はインタフェース32の意味的仕様および構文的仕様を表すメタデータ、32はサーバ30のインタフェースである。40はクライアントとサーバ間のインタフェースの変換マッピングを生成するインタフェース変換サーバ、41は前記生成した変換マッピングを参照する代理サーバ50を生成する代理サーバ生成部である。50は代理サーバ、51は代理サーバのインタフェース、52はサーバを呼び出すサーバ呼び出し部である。60は変換テーブルであり、インタフェース変換サーバ40が生成した、クライアントとサーバ間のインタフェース変換マッピングを登録する。70はオブジェクトリクエストブローカ(ORB)であり、分散オブジェクトシステムの通信基盤となるCORBA仕様で定義される。

【0013】図2は、インタフェース変換サーバが代理サーバを生成するときの処理フローを説明する図である。また、図3は図2に示す処理フローに伴うデータの流れを説明する図である。以下これらの図を参照してインタフェース変換サーバが代理サーバを生成するときの処理フローを説明する。なお、図3において、210はクライアントおよびサーバにおけるメタデータの意味的仕様を表すデータ、220はクライアントにおけるメタデータの構文的仕様を表すデータ、230はサーバにおけるメタデータの構文的仕様を表すデータである。また、図3において、図2の各ステップにおけるデータの

流れには図2と同一符号を付してその説明を省略する。

【0014】ステップ101において、クライアント10はサーバ名称あるいは提供サービス名称を検索サーバに送信して、クライアントが利用したいサービスを提供しているサーバのリファレンスを問い合わせる。ステップ102において、クライアントは前記問い合わせの回答を取得すると取得したサーバのリファレンスを用いて当該サーバ、例えばサーバ30に接続してサーバ30にクライアント10のメタデータ11を渡す。なお、全てのサーバは該サーバにクライアントがメタデータを渡すことのできるインタフェースを備えている。ステップ103において、サーバ30はクライアント10から受け取ったクライアント10のメタデータ11およびサーバ30のメタデータ31をインタフェース変換サーバ40に渡す。ステップ104において、インタフェース変換サーバ40は受け取ったメタデータ11およびメタデータ31を用いてサーバ30とクライアント10間におけるインタフェース変換マッピング生成し、生成したマッピングを変換テーブル60に登録する。

【0015】ステップ105において、インタフェース変換サーバ40の代理サーバ生成部41は、クライアント10において呼び出し可能な代理サーバ50を生成する。ステップ106において、前記生成した代理サーバ50のリファレンスをクライアント10に返す。すでに前記代理サーバ50と同等の変換を実施できる代理サーバが生成されている場合には、新たな代理サーバを生成することなく、既存の代理サーバのリファレンスをクライアントに返すこともできる。

【0016】図4は、クライアントが有するメタデータ11およびサーバが有するメタデータ31に含まれる意味的仕様の例を示す図である。意味的仕様210は、例えば銀行業務の各操作を階層構造に分類し、口座への預入操作の記述を{銀行業務/口座管理/預入}とし、その入力は{銀行業務/口座管理/預入/入力/口座番号}、および{銀行業務/口座管理/預入/入力/預入額}であり、その出力は{銀行業務/口座管理/預入/出力/残高}である。このように銀行の口座の意味的仕様について形式的な定義を与えることで、構文的仕様の対応などについて検証することができる。なお、クライアントが有するメタデータ11に含まれる意味的仕様とサーバが有するメタデータ31に含まれる意味的仕様は同一である。

【0017】図5は、クライアントが有するメタデータ11に含まれる構文的仕様の例を示す図である。構文的仕様220は関数の名称、関数の戻り値の型、引数の並び(引数の名称、型、入力、出力、入出力の指定)などからなる。例えば、銀行口座への預入操作の場合、{int deposit(in int id, in uint amount)}という記述になる。また、関数depositに対する意味として、意味的仕様210

の要素 {銀行業務211/口座管理212/預入213} が対応し、第1引数idに対する意味として {銀行業務211/口座管理212/預入213/入力214/口座番号215} が対応し、第2引数amountに対する意味として {銀行業務211/口座管理212/預入213/入力214/預入額216} が対応し、戻り値の意味として {銀行業務211/口座管理212/預入213/出力217/残高218} が対応する。

【0018】図6は、サーバ30が有するメタデータ31に含まれる構文的仕様を示す図である。構文的仕様230は図5に示すクライアントのそれと同様に、銀行口座への預入操作は {void yokin(out int zandaka, in int kouza)} となる。関数yokinは意味的仕様210の {銀行業務211/口座管理212/預入213} に対応し、第1引数zandakaは {銀行業務211/口座管理212/預入213/出力217/残高218} に対応し、第2引数kingakuは {銀行業務211/口座管理212/預入213/入力214/預入額216} に対応し、第3引数kouzaは {銀行業務211/口座管理212/預入213/入力214/口座番号215} に対応する。

【0019】図7はメタデータ11およびメタデータ31からクライアント10とサーバ30間のインタフェース変換マッピングを生成する処理を示す図である。まず、ステップ111において、クライアントの構文的仕様220の全ての要素について探索が終了したか否かを判断する。全ての探索が終了していれば処理を終了する。探索が終了していなければステップ112に進む。ステップ112において、クライアントの構文的仕様220の要素のうち探索すべき要素の一つを決定し、その要素が持つ意味と同一の意味を有するサーバの構文的仕様220の要素を探索する。ステップ113において、同一の意味を有するサーバの構文的仕様220の要素を見つけると、サーバとクライアント間の引数および戻り値の対応付けを決定する。ステップ114において、決定した対応付けを変換テーブル60に登録してステップ111に戻る。

【0020】この手順を続行することによって、例えば、構文的仕様220の要素が有する意味の {銀行業務/口座管理/預入/入力/口座番号} からは、クライアント10の呼び出す関数depositの第2引数amountと関数yokinの第2引数kingakuの対応を、{銀行業務/口座管理/預入/出力/残高} からは関数depositの戻り値と関数yokinの第1引数zandakaの対応を求めることができる。

【0021】図8は図1に示す変換テーブル60の内容を示す図である。図において301はサーバが備えるインタフェースの構文的仕様を表すサーバインタフェース、302はクライアントが備えるインタフェースの構

文的仕様を表すクライアントインタフェースである。303はサーバリファレンスであり、代理サーバ50に呼び出されて実際の処理を行うサーバのリファレンスである。304はインタフェース変換マッピングであり、代理サーバ50はインタフェース変換マッピング304を用いて、インタフェース51を介して受け付けたリクエストをインタフェース32を介してサーバ30を呼び出す信号に変換する。このインタフェース変換マッピング304は前記図2のステップ104に示す例においては、関数depositの第1引数idは関数yokinの第3引数kouzaに、depositの第2引数amountはyokinの第2引数kingakuに、またdepositの戻り値はyokinの第1引数zandakaにそれぞれ対応付けて変換する。また、305は代理サーバのリファレンスである。代理サーバが備えるインタフェースはクライアントが備えるクライアントインタフェース302と同一である。

【0022】図9は代理サーバ利用するときの処理を示すフローである。まず、ステップ401において、クライアントは図2に示すステップ106の処理により取得した代理サーバ50のリファレンスを用いて代理サーバ50のインタフェース51を呼び出す。ステップ402において、代理サーバ50は変換テーブル60に登録されているインタフェース変換マッピング304を参照して、クライアント10から受け付けたリクエストを、サーバ30のインタフェース32へのリクエストに変換し、CORBA仕様に記載されたDIIを用いてサーバ30を呼び出す。

【0023】このように、クライアントが呼び出しに用いるインタフェースとサーバがリクエストを受け付けるインタフェースとを実行時に動的に変換するので、クライアントおよびサーバのインタフェースを固定することなく柔軟に変更することができる。

【0024】図10は、本発明の第2の実施形態を示す図であり、インタフェース変換サーバが代理サーバを生成するときの他の処理フローを説明する図である。まず、ステップ201において、クライアントは10はサーバ名称あるいは提供サービス名称を検索サーバに送信して、クライアントが利用したいサービスを提供しているサーバのリファレンスを問い合わせる。問い合わせの回答を取得するとクライアントは取得したサーバのリファレンスを用いて当該サーバ、例えばサーバ30に接続してサーバ30にクライアント10のメタデータ11を渡す。このときメタデータ501をサーバ30には解読不能であるがインタフェース変換サーバ40には解読可能に暗号化して暗号化メタデータ501として送信する。ステップ202において、サーバ30はクライアント10から受け取ったクライアント10の暗号化したメタデータ501およびサーバ30のメタデータ31をインタフェース変換サーバ40に渡す。インタフェース変

換サーバ40は受け取ったメタデータ501を解読し、さらに解読したメタデータ11およびメタデータ31を用いてサーバ30とクライアント10間におけるインタフェース変換マッピング生成し、生成したマッピングを変換テーブル60に登録する。以降の処理は図2に示す処理フロー同一であるので説明を省略する。

【0025】本実施形態によれば、暗号化したメタデータ501はサーバ30で解読不可能であるが、インタフェース変換サーバ40では解読可能である。したがって、クライアント10は、サーバ30にクライアント10が呼び出しているインタフェースを知られることなくサーバ30を利用することができる。また、クライアント10はサーバ30の持つインタフェース32を未知のままサーバ30を呼び出すことができる。このように、サーバとクライアントがお互いの具体的なインタフェースを知ることなく呼び出しが可能であるため、具体的なインタフェースを知られることによるセキュリティ上の弱点が発見される可能性が低減する。

【0026】以上説明したように、本実施形態によれば、クライアントが呼び出しに利用するインタフェースとサーバがリクエストを受け付けるインタフェースを実行時に動的に変換することで、クライアントが利用するインタフェースを既に定義済みのインタフェースに固定せずに柔軟に変更でき、しかも通常とは異なる呼び出し方法でクライアントを作る必要もない。

【0027】また、実行時の具体的なインタフェースを利用者とサービス提供側がお互い知らなくてもサービスを利用可能であるので、具体的なインタフェースを知られることによりセキュリティ上の弱点を発見される可能性を低減できる。また、なんらかの理由によりインタフェースの開示ができないときもサービスの利用が可能となる。

【0028】

【発明の効果】以上説明したように本発明によれば、クライアントが利用するインタフェースを定義済みのインタフェース以外に変更することができ、また、実行時の具体的なインタフェースを利用者およびサービス提供側

が互いに未知の状態で通信可能な分散オブジェクトの連携装置を提供することができる。

【図面の簡単な説明】

【図1】本発明の第1の実施形態にかかる分散オブジェクトの連携装置を示す図である。

【図2】インタフェース変換サーバが代理サーバを生成するときの処理フローを説明する図である。

【図3】図2に示す処理フローに伴うデータの流れを説明する図である。

【図4】サーバおよびクライアントの有するメタデータに含まれる意味的仕様の例を示す図である。

【図5】クライアントの有するメタデータに含まれる構文的仕様の例を示す図である。

【図6】サーバの有するメタデータに含まれる構文的仕様を示す図である。

【図7】クライアントとサーバ間のインタフェース変換マッピングを生成する処理を示す図である。

【図8】変換テーブルの内容を示す図である。

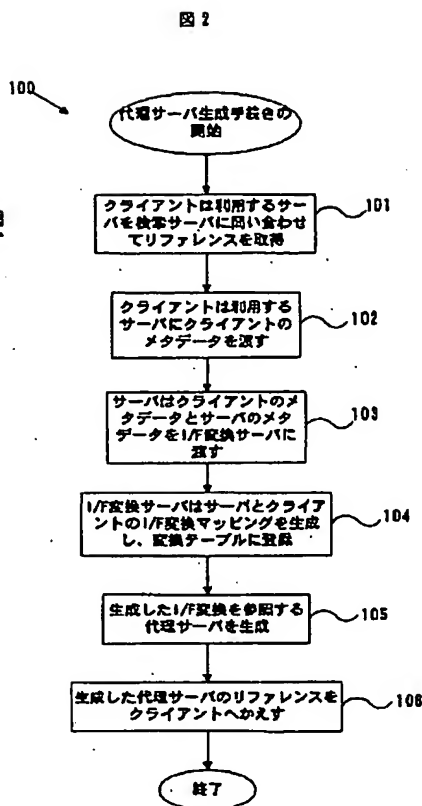
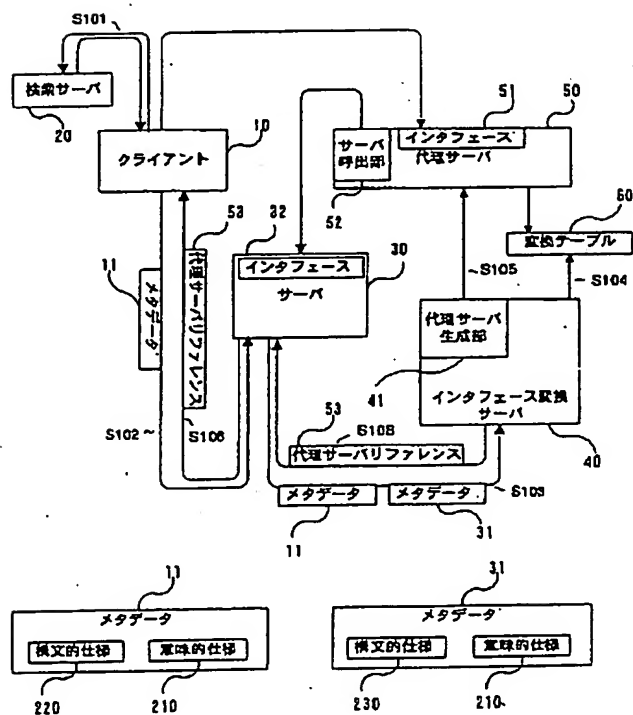
【図9】代理サーバ利用するときの処理を示すフローである。

【図10】本発明の第2の実施形態を示す図である。

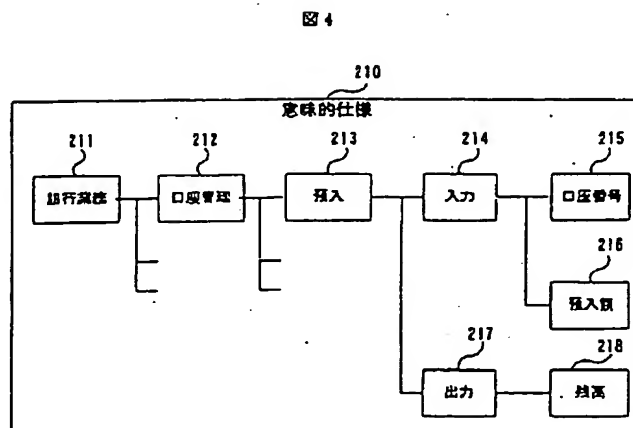
【符号の説明】

- 10 クライアント
- 12 クライアントのメタデータ
- 20 検索サーバ
- 21 リファレンス検索部
- 22 リファレンスデータベース
- 30 サーバ
- 31 サーバのメタデータ
- 32 サーバのインタフェース
- 40 インタフェース変換サーバ
- 41 代理サーバ生成部
- 50 代理サーバ
- 51 代理サーバのインタフェース
- 52 サーバ呼び出し部
- 60 変換テーブル
- 70 オブジェクトリクエストブローカ

【図2】

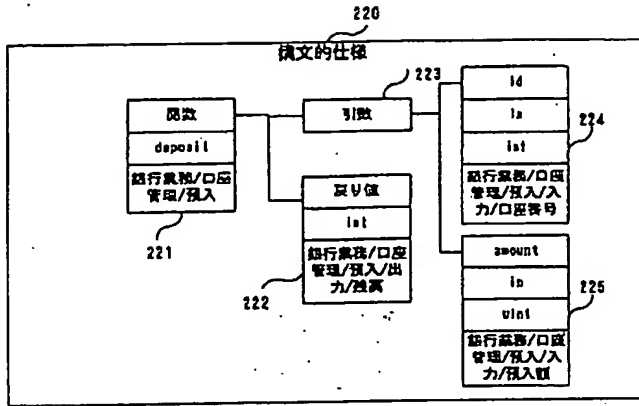
**■ 3**

【図 4】



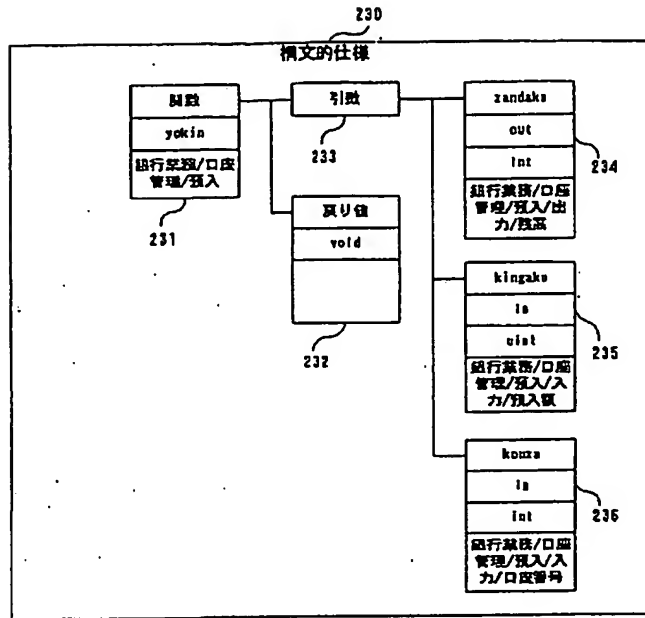
【図5】

図5



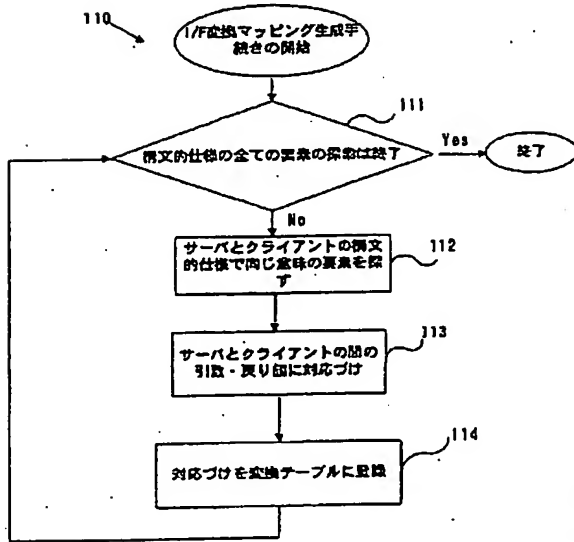
【図6】

図6



【図7】

図7



【図8】

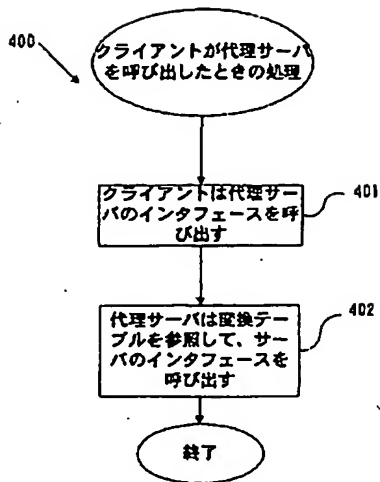
図8

図8は、交換テーブル60の図示。60は、交換テーブル。60は、サーバインタフェース301、クライアントインタフェース302、サーバリファレンス303、インタフェース変換マッピング304、代理サーバリファレンス305を含む。

サーバインタフェース	クライアントインタフェース	サーバリファレンス	インタフェース変換マッピング	代理サーバリファレンス
構文的仕様S deposit(...)	構文的仕様C yokin(...)	objref_S...	id->kouza, amount->kingaku...	objref_P...
...
...

【図9】

図9



【図10】

図10

